Xu, Jie; Capretz, Luiz Fernando; Ho, Danny

Article

An empirical process to derive OSS defect estimation models

The International Journal of Management Science and Information Technology (IJMSIT)

Provided in Cooperation with:
North American Institute of Science and Information Technology (NAISIT), Toronto

Suggested Citation: Xu, Jie; Capretz, Luiz Fernando; Ho, Danny (2011) : An empirical process to derive OSS defect estimation models, The International Journal of Management Science and Information Technology (IJMSIT), ISSN 1923-0273, NAISIT Publishers, Toronto, Iss. 1-(Jul-Sep), pp. 1-26

An official publication of The North American
Institute of Science and Information Technology

ISSN:1923-0265

INTERNATIONAL JOURNAL OF

Management Science and Information Technology

NAISIT
PUBLISHERS

www.naisit.org

The International Journal of Management Science and Information Technology (IJMSIT)

NAISIT Publishers

The International Journal of Management Science and Information Technology (IJMSIT)

NAISIT Publishers

Issue 1 - (Jul-Sep 2011)

Table of Contents

This is one paper of
The International Journal of Management Science and Information Technology (IJMSIT)
Issue 1 - (Jul-Sep 2011)

NAIST North American Institute of Science and Information Technology

The International Journal of Management Science and Information Technology (IJMSIT)
Issue 1 - (Jul-Sep 2011) (1 - 26)

AN EMPIRICAL PROCESS TO DERIVE OSS DEFECT ESTIMATION MODELS

JIE XU[1], LUIZ FERNANDO CAPRETZ[1], AND DANNY HO[2]

[1]DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, UNIVERSITY OF WESTERN ONTARIO, LONDON, CANADA

[2]NFA ESTIMATION INC., RICHMOND HILL, CANADA

Abstract

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Quality management in Open Source Software (OSS) has become a heated topic since the open source development model emerged. Much work has been done on exploring the distinct quality attributes in OSS, but very few studies covered quality estimation. In this paper, a general procedure is proposed to derive software quality estimation models for OSS projects and various candidate techniques are suggested for individual steps. The purpose is to build a model that estimates the number of defects in a project. Several statistical techniques and a machine learning approach are used to examine the significance of quality predictors. Moreover, a neuro-fuzzy approach is adopted to improve accuracy of the estimation model. This procedure is followed and validated based on data from OSS projects.

Keywords:

software quality, quality estimation, software metrics, regression, neurazl networks, fuzzy logic

1 INTRODUCTION

Software quality assurance is vital in software project management. Studies on many aspects of software quality activities were completed in the last few decades and many conclusions have been drawn on methods to improve software quality. One research interest in this area is to establish software quality estimation models that can be used for quality estimation at the early stages of projects. The estimation results can act as guidelines to enhance the quality assurance performance.

Many characteristics of software quality have been discussed to elaborate its meaning and quality estimation models have been proposed to provide predictions for various quality attributes. The estimation targets of those models can be categorized into three groups: defect content, fault-proneness and reliability. Defect content estimation models are designed for estimating the number of defects in the project, or along with the number of field defects (Bibi et al, 2006; Chulani & Boehm, 1999; Fenton & Neil, 1999; Khoshgoftaar & Gao, 2007) . Fault-proneness estimation models are used to identify classes that are fault-prone (Hochman et al, 1996; Takahashi et al 1997; Succi et al, 2003; Yu et al, 2002). The most popular reliability estimation models are Software Reliability Growth Models (SRGMs), which are a group of models for estimating

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Mean-Time-To-Failure (MTTF) or the number of failures in a stated time interval (Jelinski & Moranda, 1972; Goel & Okumoto, 1979; Littlewood, 1981; Musa & Okumoto, 1984). Although the inherent features of the models are diverse, the techniques adopted for building the models are mostly statistical methods and soft computing.

For Open Source Software (OSS), there are relatively fewer studies on quality estimation models, although many analyses were done on the quality practices in this domain (Aberdour, 2007). Koch and Neumann (2008) collected data points from hundreds of OSS projects and explored the correlations among process metrics, as well as product metrics and faulty classes. The results were presented both at class level and project level, but only qualitative comparisons were performed. Another study adopted a defect content estimation approach in the closed-source environment, which used object-oriented design metrics as predictors to estimate the number of defects in modules (Gyimothy et al, 2005).

Theoretically, the approaches that have been adopted to build quality estimation models in the closed-source environment can be replicated for OSS, but they are mostly not validated due to the distinct development style of OSS projects.

In this study, we propose a general procedure to establish quality estimation models and validate its applicability using data from OSS projects. The open source community is more open and willing to share project information with the public. Therefore, it is a good choice to collect OSS data for the empirical study.

Two research questions are tackled:

(1) Is the proposed procedure effective to derive software quality estimation models?

(2) Are the suggested techniques practical to accomplish the tasks in individual steps?

The remainder of this paper is organized as follows. In Section 2, a general modeling procedure is proposed to establish quality estimation models and effective techniques are suggested to perform the tasks in the corresponding steps. Empirical results are analyzed to validate the procedure in Section 3. Finally, we summarize the conclusions and future work in Section 4.

2 MODELING METHODOLOGY

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

We recommend a general procedure to deal with modeling problems in software quality estimation. Although prior studies on quality estimation have performed some steps or techniques, very few of them have addressed explicitly the procedure in deriving the model. The six steps in this procedure are displayed in Figure 1. The following sub-sections discuss the steps and their related techniques.

FIGURE 1: A general procedure to build estimation models

2.1 Literature Review

In this step, the first goal is to determine the target of quality estimation, i.e. what kind of quality estimation model is supposed to be built. No particular skill is required in this step, except complete understanding of the estimation problems in this area. After reviewing papers and books in related areas, we focus on quality predictors at the project level to estimate the number of defects in the project and possible techniques for building the model. In current practices of software project management, one of the key issues is to trace, fix and manage defects.

2.2 Data Preparation

Historical data of previous software projects is necessary for building a quality estimation model. Two common methods are available for collecting historical data: 1) obtain data from some data repositories; however, data is very limited in this area and is neither tailored nor sufficient for specific research purpose; 2) collect data from scratch by sending questionnaires or online surveys. The latter method is appropriate for customized data requests, but it will be diffi-

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

cult to attract a great number of respondents. After the data is collected, redundant and unrelated information need to be filtered based on preliminary analysis. Moreover, data transformation may be necessary to enable further processing.

2.3 Metrics Validation

Since information collection is based on heuristic knowledge or expertise, not all of the collected items are appropriate for quality estimation. The significance of the metrics has to be validated to include the most suitable ones for the estimation model. Correlation analysis, ANOVA and machine learning are candidate techniques for choosing the metrics. The results should be examined carefully to retain all effective predictors.

Although the correlation between two variables does not necessarily result in causal effect, it is still an effective method to choose candidate metrics. Since not all of the quantitative explanatory variables follow normal distributions, Spearman's rank correlation coefficient method is chosen instead of standard correlation coefficient methods (Hogg & Craig, 1995).

For the qualitative predictors, one way Analysis of Variance (ANOVA) can be performed to find out how effective they are for the dependent variable. To avoid the assumption of a normal distribution and to make the results more robust, ranks of the dependent variable (from the lowest to the highest) can replace the actual values in the formula to calculate the test statistic of Kruskal-Wallis ANOVA (Kruskal & Wallis, 1952).

Knowledge discovery techniques in the area of machine learning are also suitable for determining the significant predictors. This is called feature selection to fulfill the objective of dimension reduction in data mining. In this paper, we suggest the use of a modified regression tree (model tree) to explore the relationships between the quality predictors and response variable (Wang & Witten, 1997). The structure of the derived tree illustrates the relationships and the significance of the predictors can be determined.

2.4 Statistical Modeling

For building algorithmic estimation models based on historical data from previous projects, statistical modeling is the most popular approach. For the qualitative variables, traditional regression approach cannot be performed directly with the original coding of the categorical levels. In other studies, the ratings of the qualitative variables are assigned by experts and subject to

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

introduction of uncertainty into the model. In this paper, we try to determine the ratings directly from the data. In this step, the metrics can also be validated during the regression process. This is illustrated by an arrow feeding back to the previous step (as shown in Figure 1).

Ordinary linear regression is not applicable in this case because many predicators are qualitative. A special approach named CATREG (Categorical regression with optimal scaling using alternating least squares) obtains the final regression formula by assigning numerical values to the qualitative variables. The rationale behind it is to transform the categorical variables according to the optimal scaling levels (nominal or ordinal) and optimize the quantifications following the least square criterion (Van der Kooij & Meulman, 1997).

The CATREG algorithm is executed on the transformed values of the data sets. For numeric variables, a linear transformation is made. Consequently, CATREG is equivalent to a standard linear regression when the qualitative predictors are substituted by the transformed values (optimal scaling). As a result, traditional regression techniques like stepwise linear regression can be applied by assigning the obtained optimal scaling values to the qualitative independent variables (Angelis et al, 2001).

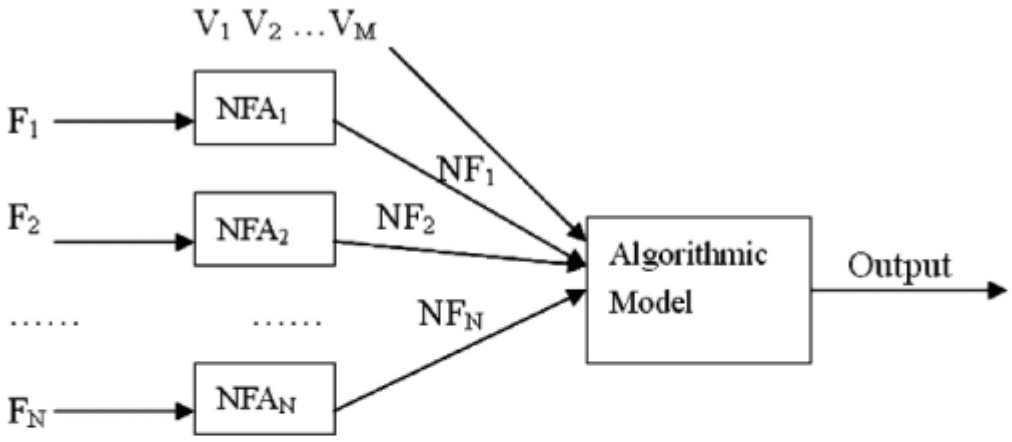
2.5 Model Recalibration

The core problem here is the suitability of the quantifications of the qualitative predictors, since the ratings used in the statistical modeling are from expert opinions, regression or other techniques. These values are usually subjective or imprecise. Therefore, soft computing techniques can be applied to further calibrate the parameters in the models. A neuro-fuzzy approach is proposed to improve the estimation accuracy by combining the advantages from fuzzy logic to handle imprecise inputs and the learning capability of neural networks.

In this paper, we adopt the Adaptive Neuro-Fuzzy Inference System (ANFIS) to further adjust the quantifications in the derived regression model (Jang et al, 1997). The purpose of embedding ANFIS in the recalibration structure is to fine-tune the quantifications of those qualitative variables in order to achieve better estimation performance. The actual structure of the recalibration approach is shown in Figure 2. Each of the neuro-fuzzy analysis (NFA) unit is a single ANFIS for each of the qualitative variable and the quantitative variables are fed directly into the algorithmic model without recalibration (Huang et al, 2006; Huang et al, 2007). The training process can be carried out using a back-propagation method. During the process, the error

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

signals are fed back to each NFA and the parameters of the NFA are then adjusted following the gradient-based algorithm. When the training process converges, the trained consequent parameters of each NFA are regarded as baselines of the final quantifications of that qualitative variable.



where F_i is the qualitative variable

NF_i is the fine-tuned qualitative variable

NFA_i is the neuro-fuzzy analysis unit

V_j is the quantitative variable

FIGURE 2: Structure of the recalibration approach

2.6 Model Evaluation

The model derived from the previous steps has to be evaluated according to certain criteria to compare the performance. Many criteria exist for assessing the effectiveness of the estimation models, such as MMRE and *Pred(m),* yet there is not an unanimously accepted criterion to judge the performance. Also, cross-validation is commonly adopted to avoid the overfitting problem. *K*-fold cross-validation is the most commonly utilized method, in which the dataset is divided into *k* subsamples. One subsample is reserved as validation (testing) data, while the remaining *k*-1 subsamples are used as training data for building the model. The cross-validation process is therefore to be repeated *k* times and the *k* results are averaged to determine the final performance of the model.

3 Empirical Results

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

We collect project information from the open source community. Then, our proposed procedure is performed based on the acquired data to validate its effectiveness.

3.1 Data Source

SourceForge.net is one of the largest OSS development website in the world. Because of its popularity, we chose project data from SourceForge.net to conduct our exploratory analysis. Certain SourceForge.net data has been shared with the University of Notre Dame for research purposes and it consists of more than 100 tables in the data dumps. On the other hand, a project named FLOSSmole (Collaborative collection and analysis of free/libre/open source project data) has been developed to supply data from OSS projects to the public domain (Howison et al. 2006). Web crawling of the most popular OSS hosts, including SourceForge, Rubyforge, Fresh-meat, etc., has been performed mostly on a monthly basis to collect data from these websites. We only focus on OSS projects hosted on SourceForge.

3.2 Experimental Results

3.2.1 Data Preparation

Since there are too many OSS projects on SourceForge.net, we have to develop criteria to select suitable ones for our research. Projects that have been developed with attention to their quality characteristics and have attained certain development status are being considered. We suggest two factors to judge the popularity of the software: download count and rank. We mainly focus on matured OSS projects. Approximately 1500 OSS projects from SourceForge have been selected and there are sufficient data points for building the estimation model.

A questionnaire was designed and sent to the administrators of the selected OSS projects, in order to obtain other essential project information. To simplify the process, the questionnaire only comprised of 22 multiple choice questions (see Appendix). These questions took into account the project plan and design, quality requirement, personnel, product complexity, testing and related tools, documentation and so on. Some questions about product complexity were adopted from those of COQUALMO (Chulani, 1999). Using the registered names of the project administrators gathered from FLOSSmole, an email list was developed for distribution of the questionnaire. We used direct email contact instead of online survey because the latter was difficult to draw the attention of the target group. Only 278 valid responses were received and many of the sent emails might have been treated as junk emails and ignored.

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Another critical software attribute of the OSS projects is size. It seems impossible for the OSS project administrators to keep record of the product size, even in the simplest form of source line of code (SLOC). Moreover, the OSS projects usually involve several programming languages. It is totally unreasonable to merely sum up the counts of SLOC of all languages. Therefore, we count logical lines of code for different languages first and then utilize the backfiring method to derive function points (Wong et al, 2008).

We integrated all data together for later experiments. The data composed of responses from the questionnaire (22 questions), size metric from backfiring, duration, team size and defect information from web crawling. Finally, 194 OSS projects were kept for further experiments.

3.2.2 Metric Validation

3.2.2.1 Correlation Analysis

We considered some of the variables, Function Points (size), Duration and Team Size as numeric and analyzed the correlation with the number of defects in the project.

TABLE 1: Correlation Results of the OSS Data

Factors	Correlation coefficient	Sig. (2 tailed)
Function points	0.470	**0.000**
Duration	0.373	**0.000**
Team size	0.213	**0.003**

From the results of correlation analysis (Table 1), Function Points, Duration and Team Size all showed some correlation with the number of defects at the 0.05 significance level, but the correlation coefficients were not high. Therefore, we decided to include these variables in the following regression modeling, but their influences needed to be further examined.

3.2.2.2 ANOVA

Kruskal-Wallis ANOVA is used to examine the effects that these qualitative variables had on the number of defects (the dependent variable). From the results listed in Table 2, we conclude that Q2, Q3, Q4, Q9, Q 10, Q11, Q17, Q18, Q20, Q21 and Q22 seem to have significant influences on the dependent variable, i.e. the number of defects.

TABLE 2: ANOVA results of the OSS data

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Predictors	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Sig.	0.265	**0.043**	**0.014**	**0.027**	0.508	0.353	0.493	0.861	**0.049**	**0.001**	**0.000**
Predictors	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
Sig.	0.113	0.058	0.097	0.181	0.445	**0.033**	**0.012**	0.098	**0.021**	**0.040**	**0.034**

3.2.2.3 Machine Learning

We continue to use a model tree to explore the relationships between the dependent variable (defects) and predictors (both quantitative and qualitative). The candidate predictors are the 22 questions, Function Points, Duration, Team Size, etc. There is only one leaf (branch) in the derived model tree, where no splitting attribute exists. That means it is not applicable to separate the data points into various sub-samples. When the formula at the leaf is examined, we identify Q2, Q3, Q9, Q10, Q11, Q17, Q18, Q20, Q22, Function Points and Duration as independent variables. The results are consistent with those in the previous steps.

3.2.3 Regression

The numeric variables under investigation are the number of defects, Function Points and Duration. Natural logarithmic transformation is applied to make them follow the normal distribution required by later regression. Firstly, we apply CATREG to the selected variables in order to obtain the quantifications of the qualitative variables. The optimal scaling level of quantitative variables is set to numeric and that of qualitative variables is defined as ordinal due to their characteristics. The model summary is listed in Table 3, which illustrates R-square as 0.556.

TABLE 3: Model summary of first-round CATREG

	Multiple R	R Square	Adjusted R Square
Standardized Data	.745	.556	.495
Dependent Variable: Defects Predictors: Q2 Q3 Q9 Q10 Q11 Q17 Q18 Q20 Q22 FP Duration			

Although the model is overall significant (Sig.<0.05), we still have to examine the effectiveness of each predictor. The coefficients of the regression are omitted for simplicity. We discover

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

that not all of the coefficients are significant at the 0.05 level, which means that the respective coefficients may not be statistically valid estimates.

The stepwise linear regression is executed based on the quantifications achieved by CA-TREG, in order to make sure what predictors can be entered into the regression model. The results of the stepwise linear regression are listed in Table 4 (model summary, showing R-Square in the stepwise steps) and Table 5 (showing regression coefficients in the stepwise steps). Model 9 was the final model chosen by the stepwise linear regression and the results of the previous stepwise steps were excluded.

TABLE 4: Model summary of first-round stepwise regression

Model	R	R Square	Adjusted R Square
9	.739[i]	.545	.523
[i]Predictors: (Constant), FP, Q10 Quantification, Q11 Quantification, Duration, Q3 Quantification, Q18 Quantification, Q9 Quantification, Q2 Quantification, Q17 Quantification			

TABLE 5: Coefficients of first-round stepwise regression

Model		Unstandardized Coefficients	Standardized Coefficients	Sig.
9	(Constant)	-2.638		.001
	FP	.453	.516	.000
	Q10 Quantification	.294	.217	.001
	Q11 Quantification	.231	.171	.004
	Duration	.448	.216	.000
	Q3 Quantification	-.182	-.135	.013
	Q18 Quantification	.153	.113	.028
	Q9 Quantification	.190	.141	.019
	Q2 Quantification	.191	.141	.007
	Q17 Quantification	.147	.108	.035

The independent variables that enter into the regression model are Q2, Q3, Q9, Q10, Q11, Q17, Q18, FP and Duration. Since CATREG may result in another set of quantifications with different number of variables in the regression process, we have to perform CATREG once again, but only including the above independent variables and the dependent variable. The model summary of the second-round CATREG is listed in Table 6. It shows that the R-square is 0.548,

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

which is a little lower than the first round due to fewer predictors. The coefficients of the second-round CATREG are not presented, but this time only the coefficient of Q17 seems not significant at the 0.05 level.

TABLE 6: Model summary of second-round CATREG

	Multiple R	R Square	Adjusted R Square
Standardized Data	.740	.548	.501
Dependent Variable: Defects Predictors: Q2 Q3 Q9 Q10 Q11 Q17 Q18 FP Duration			

Based on the new quantifications achieved by CATREG, the stepwise linear regression can be executed again in order to confirm what predictors should be included into the regression model. The results of the second-round stepwise linear regression are listed in Table 7 (model summary) and Table 8 (showing regression coefficients in the stepwise steps). Similarly, Model 9 is the final model chosen by the stepwise linear regression and the results of the previous step-wise steps are omitted.

TABLE 7: Model summary of second-round stepwise regression

Model	R	R Square	Adjusted R Square
9	.740[i]	.548	.526
[i] Predictors: (Constant), FP, Q10 Quantification, Q11 Quantification, Duration, Q3 Quantification, Q18-Quantification, Q2 Quantification, Q9 Quantification, Q17 Quantification			

TABLE 8: Coefficients of second-round stepwise regression

Model		Unstandardized Coefficients	Standardized Coefficients	Sig.
9	(Constant)	-2.676		.001
	FP	.460	.524	.000
	Q10 Quantification	.306	.226	.001
	Q11 Quantification	.221	.163	.007
	Duration	.446	.214	.000
	Q3 Quantification	-.179	-.132	.015
	Q18 Quantification	.152	.112	.028

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

	Q2 Quantification	.189	.140	.008
	Q9 Quantification	.188	.139	.022
	Q17 Quantification	.147	.109	.033

The results in Table 8 show that all 9 independent variables are included in the final model and all coefficients are significant at the 0.05 level.

Therefore, we derive the final regression formula with the resulting quantifications:

*Defects = 0.460*FP + 0.446*Duration + 0.189*Q2 - 0.1 79* Q3 + 0.188*Q9 + 0.306*Q10 + 0.221*Q11 + 0.147*Q17 + 0.152*Q18 – 2.676* (1)

The choices of the questions are arranged in an order from weak to strong. When we ex-amine the coefficients in the formula, only the one for Q3 is negative, which means that more experienced developers tend to produce fewer defects. For other questions such as Q2 (release frequency), Q9 (data complexity), Q10 (computational complexity), Q11 (structural complexity), Q17 (bug tracking tool) and Q18 (users involved), the defect trend follows the order of the answers. The two quantitative predictors both had positive coefficients. Therefore, bigger size results in more defects and longer duration of development leads to more defects. All the findings conform to our intuition.

3.2.4 Recalibration

Among the predictors in the previous linear regression formula, Q2, Q3, Q9, Q10, Q11, Q17 and Q18 could be regarded as qualitative variables and the derived quantifications are assigned to them. Since the quantifications are derived during the process of categorical regression, the computed values of these variables have the potential for further adjustments to make the model more accurate. Consequently, these 7 variables could continue with neurofuzzy recalibration. Thus, 7 NFA units need to be processed before entering the inputs into the derived algorithmic model. One NFA unit is established to adjust the quantifications of each qualitative variable.

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

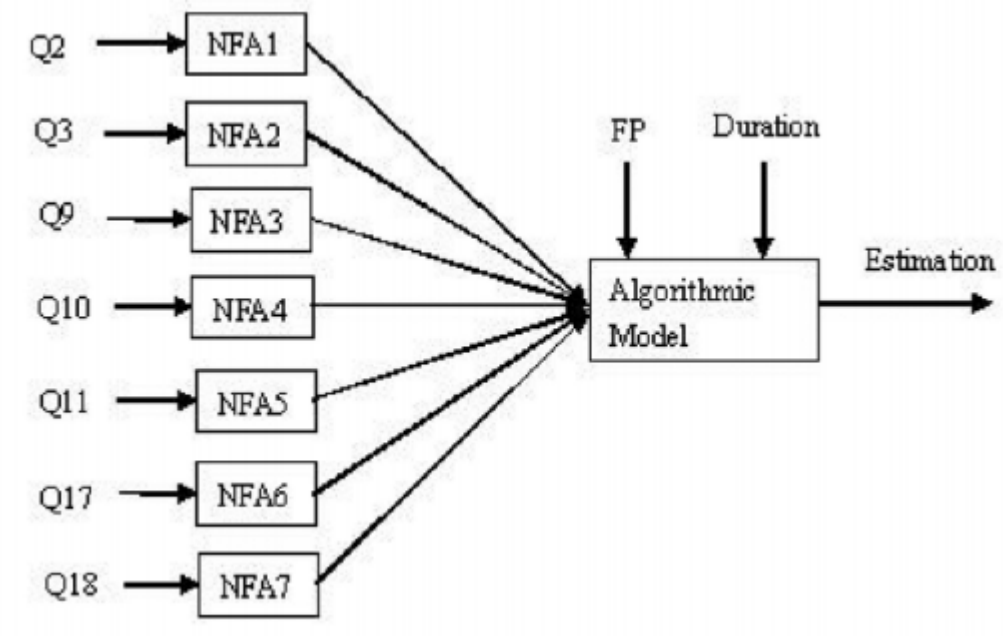


FIGURE 3: Specific structure for the recalibration

The structure of the recalibration approach is not exactly equivalent to that of a typical ANFIS, since the algorithmic model is attached to the end. Moreover, the 7 NFA units are trained simultaneously. Therefore, we only implement the gradient descent method for our application unlike the common learning algorithm of ANFIS.

For one of the consequent parameters of the fuzzy rules, for instance, c_1 is updated by:

$$\Delta c_1 = -\alpha \frac{\partial E}{\partial c_1} = -\alpha \frac{\partial E}{\partial f} \times \frac{\partial f}{\partial y} \times \frac{\partial y}{\partial (\overline{\mu}_1 c_1)} \times \frac{\partial (\overline{\mu}_1 c_1)}{\partial c_1} \qquad (2)$$

where:

f is formula of the algorithmic model (or the predicted value);

y is the output of the NFA unit;

$ì_1$ is the normalized firing strength of that fuzzy rule;

c_1 is the particular consequent parameter of that fuzzy rule.

Finally, the correction of c_1 should be calculated from (the intermediate steps are omitted):

$$\Delta c_1 = -\alpha \times \frac{1}{actual} \times \frac{(f - actual)}{|actual - f|} \times (f \cdot \beta_2) \times 1 \times \overline{\mu}_1 \qquad (3)$$

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

The training process would terminate under either of the following two conditions: the error function (objective function) converges (thus no need for further adjustment) or the checking results of the validation data subset deteriorates (the overfitting problem emerges).

3.2.5 Model Evaluation

There are 194 data points; still not sufficient when the number of adjustable parameters is considered. Therefore, cross-validation is adopted to evaluate the performance of the derived models. Both 10-fold and 6-fold cross-validations are carried out to evaluate the model before and after the recalibration using MMRE. *Pred*(m) is not comparable in this case and excluded due to great variation in the data.

We examine the improvement of the MMRE results. According to the experimental results (Table 9, 10- fold), the average MMRE is reduced from 0.9416 to 0.6504. Similarly, the 6-fold results (Table 10) present an average MMRE reduction from 0.9429 to 0.6558. Both of the improvements, i.e. 27.56% and 26. 82%, are significant.

TABLE 9: Performance of cross-validation (10-fold)

MMRE	Before	After	Improvement %
Experiment 1	0.9316	0.6433	30.95
Experiment 2	0.8457	0.6179	26.94
Experiment 3	1.5381	0.742	51.76
Experiment 4	0.7244	0.6222	14.11
Experiment 5	0.6957	0.6739	3.14
Experiment 6	0.7932	0.5346	32.61
Experiment 7	1.1899	0.807	32.18
Experiment 8	0.6492	0.5901	9.10
Experiment 9	1.1429	0.6744	41.00
Experiment 10	0.9054	0.599	33.83
Average	0.9416	0.6504	27.56

TABLE 10: Performance of cross-validation (6-fold)

MMRE	Before	After	Improvement %
Experiment 1	0.597	0.5552	7.00
Experiment 2	0.9742	0.5991	38.51
Experiment 3	0.7312	0.6206	15.12
Experiment 4	1.1968	0.6821	43.01

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Experiment 5	1.4093	0.8728	38.07
Experiment 6	0.7487	0.6048	19.22
Average	0.9429	0.6558	26.82

There might be bias during data separation. Thus we proceed with another strategy, that is, randomly choosing certain percentage of data points (50%, 60% and 75%) for training and the

remainder for checking. The respective training process is repeated 10 times for each percentage and the average values of the MMRE improvements are calculated. The respective MMRE improvements for the three cases are 29.10%, 28.78% and 26.07% (Table 11) and all of them are significant.

TABLE 11: Performance of random training

Improvement	50% Data	60% Data	75% Data
Experiment 1	31.02	34.76	25.88
Experiment 2	30.91	28.94	34.56
Experiment 3	24.92	27.83	15.76
Experiment 4	28.27	32.27	25.32
Experiment 5	31.06	25.76	24.11
Experiment 6	29.94	19.14	27.28
Experiment 7	29.79	32.78	28.63
Experiment 8	21.19	30.22	31.21
Experiment 9	31.17	29.58	17.81
Experiment 10	32.74	26.55	30.10
Average %	29.10	28.78	26.07

4 CONCLUSIONS AND FUTURE WORK

In this work we proposed a general procedure for building OSS quality estimation models. Two research questions are addressed in this paper. Firstly, the tasks of the six steps are described, which makes the general procedure more understandable and practical. Secondly, sever-al methods, such as correlation analysis, ANOVA and decision trees, are utilized for determining the software metrics for software quality estimation. Moreover, we suggest applying regression approaches like CATREG and the stepwise regression to build estimation models from data and

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

then using the neuro-fuzzy approach to recalibrate the quantifications of the qualitative parameters to improve performance. Following the proposed procedure and adopting the suggested techniques, a software quality estimation model can be derived.

We also validate the procedure step by step using the data from OSS projects. The whole procedure and the related techniques have been proved to be appropriate for deriving software quality estimation model. Due to insufficient information at hand, we cannot compare the performance of the derived model with other quality estimation models. However, the recalibration step exhibits its significance by improving the performance of the previously derived estimation model.

One highlight of the procedure is using CATREG to acquire quantifications of the qualitative predictors, which avoids subjective bias in human judgments. Another major advantage of this procedure is applying the neuro-fuzzy approach to improve estimation performance. One can first obtain a regression model from the data and then improve its performance with the recalibration approach. On the other hand, if a qualified algorithmic model already exists, the recalibration step itself can be performed to improve the estimation accuracy. This procedure is also applicable for building estimation models in other areas in software engineering.

The suggestions for future work are as follows:

(1) We prefer collection of new data points of OS S projects at their early stages. Then, defect estimation can be given by the derived model. At the end, we will be able to examine the estimation performance when the projects are completed and final defect information is available.

(2) As the only realistic data is from the open source community, we plan to develop a mechanism to acquire more quality-related information. When more predictors and more accurate defect information are available, the proposed procedure can be used to build a real tool for OSS quality estimation.

ACKNOWLEDGEMENTS

The authors would like to thank the OSS project administrators who responded to the questionnaires that provided the data for our research.

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

REFERENCES

Aberdour, M. (2007). Achieving quality in open source software. *IEEE Software,* 24(1), 5 8-64. doi: 10.1109/MS.2007.2

Angelis, L., Stamelos, I., & Morisio, M. (2001). Building a software cost estimation model based on categorical data", Software Metrics Symposium, 2001. METRICS 2001. In Proceedings. Seventh International, 4-15. doi: 10.11 09/METRIC.2001 .915511

Bibi, S., Tsoumakas, G., Stamelos, I., & Vlahavas, I. (2006). Software defect prediction using regression via classification, *IEEE International Conference on* Computer *Systems & Applications,* 330-336. Retrieved from Compendex database

Chulani, S. & Boehm, B. (1999). Modeling software defect introduction and removal: COQUALMO (COnstructive QUALity MOdel), USC-CSE Technical Report. Retrieved from http://sunset.usc.edu/csse/TECHRPTS/1999/usccse99-5 1 0/usccse99-5 1 0.pdf

Chulani, S. (1999). *Bayesian analysis of software cost and quality models,* Ph.D. Dissertation, The University of Southern California. Available from ProQuest Dissertations & Theses database. (UMI No. 9933649)

Fenton, N.E., & Neil, M. (1999). A critique of software defect prediction models, *IEEE Transactions on Software Engineering,* 25(5), 675–689. doi: 10.1109/32.815326

Goel, A.L., & Okumoto, K. (1979). A time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability,* R-28, 206-211. Retrieved from INSPEC database

Gyimothy, T., Ferenc, R., & Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering,* 31(10), 897-910. doi: 10.1109/TSE.2005.112

Hochman, R., Khoshgoftaar, T.M., Allen, E.B., & Hudepohl, J.P. (1996). Using the genetic algorithm to build optimal neural networks for fault-prone module detection, *In Proceedings: 7th International Symposium on Software Reliability Engineering,* 152-162. doi: 10.1109/ ISSRE. 1996.558759

Hogg, R.V., & Craig, A.T. (1995). *Introduction to mathematical statistics, 5th ed.* New York:

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Macmillan, 338-400.

Howison, J., Conklin, M., & Crowston, K. (2006). FLOSSmole: A collaborative repository for FLOSS research data and analyses, *International Journal of Information Technology and Web Engineering,* **1(3),** 17–26. Retrieved from http://scholar.google.com

Huang, X., Ho, D., Ren, J., & Capretz, L.F. (2006). A soft computing framework for software effort estimation, *Soft Computing - A Fusion of Foundations, Methodologies and Applications,* 10(2), 170-177. Retrieved from http://scholar.google.com

Huang, X., Ho, D., Ren, J., & Capretz, L.F. (2007). Improving the COCOMO model using a neuro-fuzzy approach, *Applied Soft Computing, 7(1),* 29-40, Elsevier Science. doi: 10.1016/j.asoc.2005.06.007

Jang, J.S.R., Sun, C.T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence,* Upper Saddle River, NJ : Prentice Hall.

Jelinski, Z., & Moranda, P. (1972). Software reliability research, *Statistical Computer Performance Evaluation,* W. Freiberger, ed., Academic Press, New York, 465-484.

Khoshgoftaar, T.M., & Gao, K. (2007). Count models for software quality estimation, *IEEE Transactions on Reliability,* 56(2), 212 – 222. doi: 10.1109/TR.2007.896757

Koch, S., & Neumann, C. (2008). Exploring the effects of process characteristics on product quality in open source software development. *Journal of Database Management,* 19(2), 31-57. Retrieved from INSPEC database

Kruskal, W.H., & Wallis, W.A. (1952). Use of ranks in one-criterion variance analysis, *Journal of the American Statistical Association,* 47(260), 583–621. Retrieved from http://scholar.google.com

Littlewood, B. (1981). Stochastic reliability growth: a model for fault removal in computer programs and hardware designs, *IEEE Transactions on Reliability,* R-30, 3 13-320. Retrieved from INSPEC database

Musa, J.D., & Okumoto, K. (1984). A logarithmic Poisson execution time model for software reliability measurement, *In Proceedings of the Seventh International Conference on Software Engineering,* Los Alamitos, Calif., 230-23 8. Retrieved from INSPEC database

Succi, G., Pedrycz, W., Stefanovic, M., & Miller, J. (2003). Practical assessment of the models

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

for identification of defect-prone classes in object-oriented commercial systems using design metrics, *Journal of Systems and Software,* 65(1), 1-12. doi: 10.1016/S0164-1212(02)00024-9

Takahashi, R., Muraoka, Y., & Nakamura, Y. (1997). Building software quality classification trees approach, experimentation, evaluation, *In Proceedings: 8th International Symposium on Software Reliability Engineering,* 222-233. doi: 10.1 109/ISSRE. 1997.630869

Van der Kooij, A.J., & Meulman, J.J. (1997). MURALS: Multiple regression and optimal scaling using alternating least squares, In: E. Faulbaum & W. Bandilla (Eds.), *Softstat '97,* Stuttgart: Lucius & Lucius, 99-106.

Wang, Y., & Witten, I.H. (1997). Induction of model trees for predicting continuous classes, *In Proceedings of the poster papers of the European Conference on Machine Learning,* Prague, Czech Republic, 128-137. Retrieved from http://scholar.google.com

Wong, J., Ho, D., & Capretz, L. F. (2008). Calibrating function point backfiring conversion ratios using neuro-fuzzy technique, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS),* **16(6),** 847-862. doi: 10.1142/S0218488508005650

Yu, P., Systa, T., & Muller, H. (2002). Predicting fault-proneness using OO metrics: an industrial case study, *In Proceedings of Sixth European Conference on Software Maintenance and Reengineering,* 99-107. doi: 10.1 109/CSMR.2002.995794

APPENDIX: OSS QUESTIONNAIRE

1. Is there a specific plan/schedule for the project?

   ( )A. No schedule

   ( )B. Somehow clear schedule

   ( )C. Very clear schedule

2. How often will the project publish new releases (on average)?

   ( )A. Not sure

   ( )B. Every year

   ( )C. Every six months

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

( )D. Every quarter

( )E. Every month

( )F. Every week

3. What is the average related software development experience of the developers? (language, application and platform)

( )A. <1 year

( )B. 1-3 years

( )C. 3-5 years

( )D. >5 years

4. What is the percentage of personnel change during the development?

( )A. <10%

( )B. 10% - 20%

( )C. 20% - 30%

( )D. 30% - 40%

( )E. >40%

5. Is there any similar project (functionality and implementation)?

( )A. None

( )B. A few

( )C. Many

6. Is there any reliability requirement for the project?

( )A. Low: Slight inconvenience or very small losses when fails;

( )B. Nominal: Moderate losses when fails;

( )C. High: High financial losses, or risk to life when fails.

7. Is there any response time constraint?

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

( )A. Low: No or loose time constraint;

( )B. Nominal: Common time constraints, no special request;

( )C. High: Strict time limit or real time system.

8. How do you deal with modularity in the project?

( )A. No consideration of modularity

( )B. Redesigned during the development stage

( )C. Prepared at the beginning of the development phase

( )D. Clearly designed during the design stage

9. What is the complexity of data management in the project?

( )A. Very low: Simple arrays; Simple DB queries, updates;

( )B. Low: Single file with no data structure changes, no edits, no intermediate files. Moderately complex DB queries, updates;

( )C. Nominal: Multi-file input and single file output. Simple structural updates. Complex DB queries, updates;

( )D. High: Simple triggers. Complex structural updates;

( )E. Very high: Distributed or complicated database management. Complex triggers. Search optimization.

10. What is the computational requirement in the project?

( )A. Very low: Only basic math expressions involved;

( )B. Low: Standard math/statistical routines needed;

( )C. Nominal: Basic numerical data analysis like ordinary differential equations and regular calculation accuracy required;

( )D. High: Complex data analysis such as partial differential equations;

( )E. Very high: Accurate numerical analysis with noisy, stochastic data.

11. What is the level of control flow in the project?

( )A. Very low: Straightforward nesting structured programming with simple decision condi-

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

tions;

( )B. Low: Basic nesting with decision tables; Simple callback and message exchange;

( )C. Nominal: Highly structured programming with complicated predicates; Queue and stack control; Basic distributed processing;

( )D. High: Recursive coding; Simple interrupt handling; Task synchronization, complex callbacks, complex distributed processing; Soft real time control;

( )E. Very high: Complex interrupt handling with changing priorities; Immediate real time control.

12. What is the requirement of user interface management?

( )A. Low: Simple forms;

( )B. Nominal: Graphic user interface;

( )C. High: 2D/3D, dynamic graphics; multimedia.

13. Do you have test plan for the project?

( )A. No test plan

( )B. Somehow clear plan (basic requirements)

( )C. Very clear test plan (test phases, test cases)

14. Do you use any tool for testing?

( )A. No

( )B.Yes (Name

15. What percentage of source code is covered during testing?

( )A. < 20%

( )B. 20% - 40%

( )C. 40% - 60%

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

( )D. 60% - 80%

( )E. > 80%

16. The previous coverage information is derived from:

( )A. Rough estimation

( )B. Coverage tool (Name

17. Is the total number of bugs recorded correctly in the Bug Tracking System?(If not, please give a number)

( )A. No (Number) (

)B.Yes

18. How many users are involved in the project?

( )A. < 5

( )B. 5 - 10

( )C. 10 - 50

( )D. 50 - 100

( )E. > 100

19. What percentage of defects/bugs do users report?

( )A. < 20%

( )B. 20% - 40%

( )C. 40% - 60%

( )D. 60% - 80%

( )E. > 80%

20. What percentage of total development effort is used for testing?

( )A. < 20%

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

( )B. 20% - 40%

( )C. 40% - 60%

( )D. 60% - 80%

( )E. > 80%

21. What documentation is used to help new developers get onboard?

( )A. No particular documentation

( )B. Major guidelines available

( )C. Detailed definition of processes and development guidelines available

22. How is the user documentation prepared?

( )A. No particular documentation

( )B. Only draft and incomplete version

( )C. Important parts covered

( )D. Detailed and comprehensive

AUTHORS

Jie Xu obtained his Ph.D. in Software Engineering at the Electrical and Computer Engineering Department, the University of Western Ontario, Canada. He currently works with Atos Origin (China) as a testing manager. His research interests include software engineering, software estimation, soft computing, and software quality.

Luiz Fernando Capretz is currently an Associate Professor and the Director of Software Engineering at the University of Western Ontario, Canada. His present research interests include software engineering (SE), human factors in SE, software estimation, software product lines, and software engineering education. Having worked in Brazil, Argentina, U.K., Japan, Italy, UAE, and Canada, he has more than 30 year of experience in the engineering of software as practitioner, manager, and educator. He is an IEEE senior member, ACM distinguished member, MBTI qualified practitioner, Professional Engineer in Ontario (Canada).

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014

Danny Ho is an independent management consultant and adviser for two startup companies. He is also appointed as an Adjunct Research Professor at the Department of Software Engineering, Faculty of Engineering, The University of Western Ontario and University of Ontario Institute of Technology. His areas of special interest include software estimation, project management, object-oriented software development, and complexity analysis. He is currently a member of the Professional Engineers Ontario (PEO) and a Project Management Professional (PMP).

ISSN 1923-0265 (Print) - ISSN 1923-0273 (Online) - ISSN 1923-0281 (CD-ROM), Copyright NAISIT Publishers 2014